



Graphics and Imaging Overview

Session 500





Graphics and Imaging Overview

Peter Graffagnino
Director, Graphics and Imaging Engineering

Agenda

- Technologies:
 - Quartz2D
 - OpenGL
 - Quartz Compositor
 - ColorSync
 - ImageCapture
 - Printing
- Brief overview of technology
- What's new in Jaguar



Aqua

Frameworks

Graphics

Darwin



Aqua

Frameworks

Quartz

OpenGL

QuickTime

Darwin



Aqua

Frameworks

Quartz

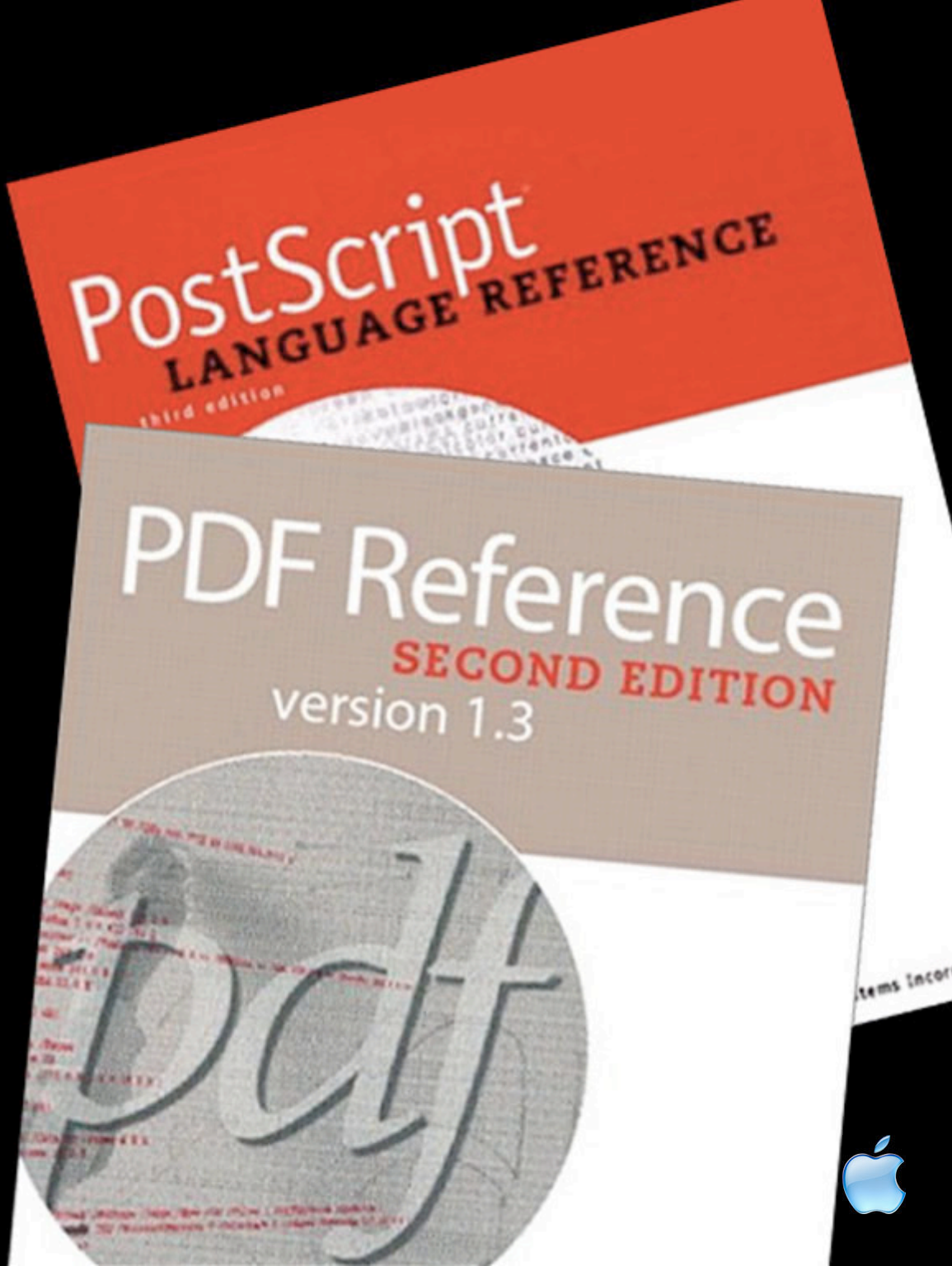
OpenGL

QuickTime

Darwin



Industry Standard 2D Imaging Model



Quartz 2D

- Lightweight C Library
- Read/Write PDF
- Fast Anti-aliasing
- Destination Alpha
- Apple Type System built-in
- Type 1 Scaler built-in
- ColorSync built-in



What's New in Quartz 2D

- Full PDF 1.3 support
- Gradients and patterns
- Transparency in PDF
- PDF X/3 support
- PICT rendering with Quartz 2D
- QD text rendering with Quartz 2D

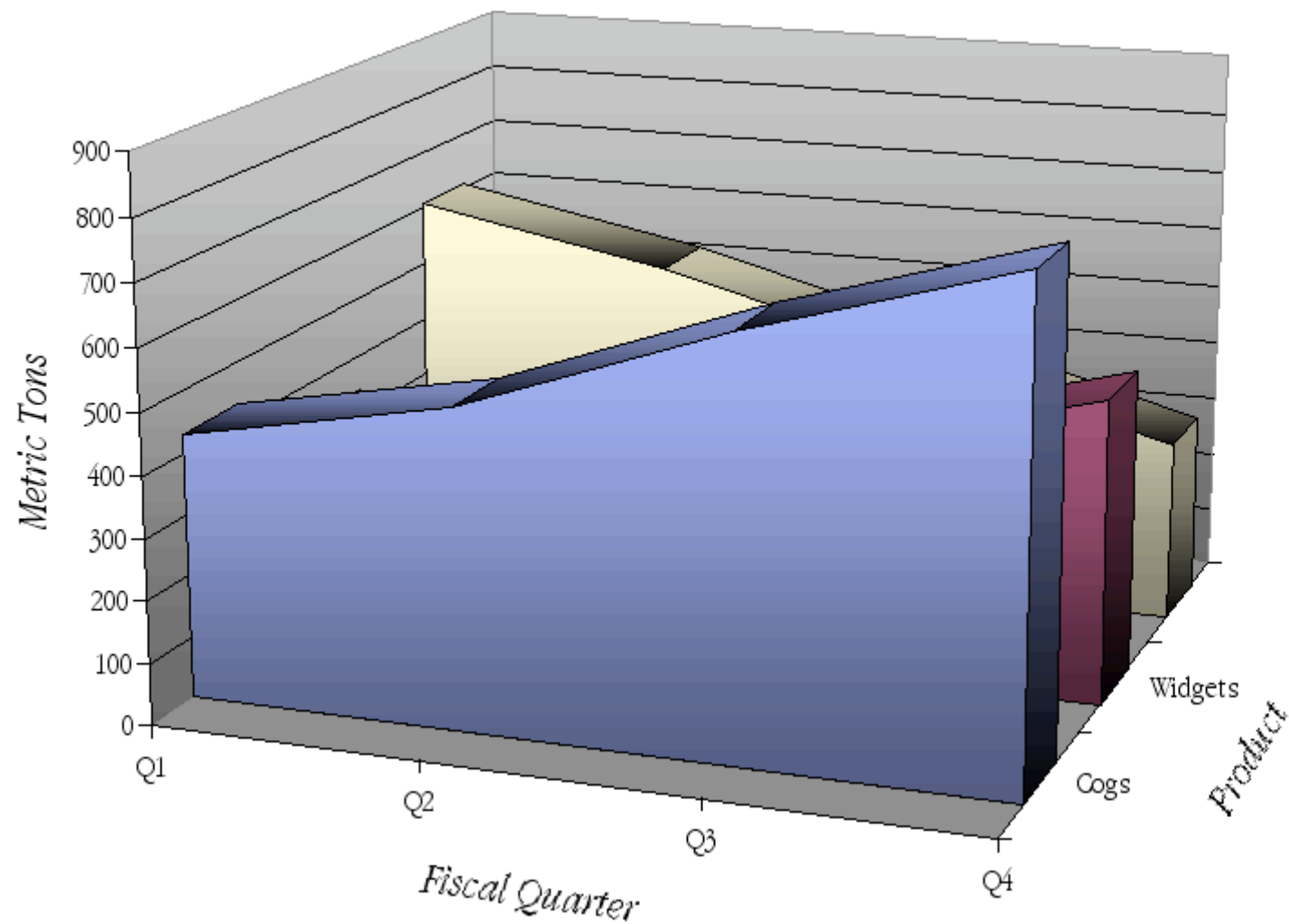


Quartz 2D and Carbon

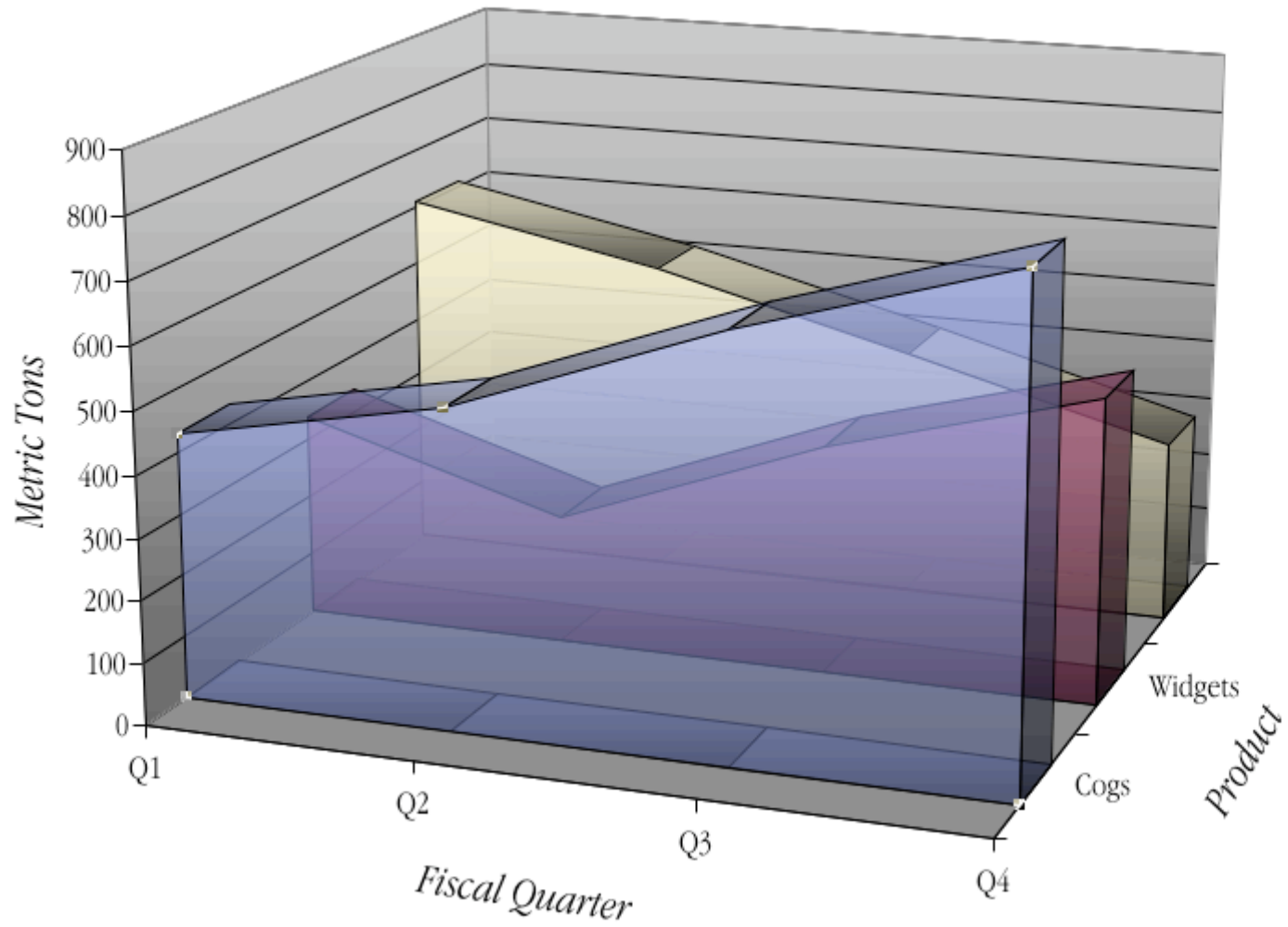
- Can mix QuickDraw and Quartz 2D



Combined Industrial Output for Acme International



Combined Industrial Output for Acme International



Moving to Quartz 2D

- PICT rendering with Quartz 2D
- QuickDraw text rendering with Quartz 2D



Quartz 2D Roadmap

501 Quartz 2D and PDF

Update on Quartz 2D APIs

Room A2
Tue., 2:00pm

509 ColorSync and Digital Media

ColorSync update

Room C
Wed., 5:00pm

516 Graphics and Imaging Performance Tuning

Hall 2
Fri., 3:30pm



Aqua

Frameworks

Quartz

OpenGL

QuickTime

Darwin



OpenGL[®] Reference Manual

Second Edition

The Official Reference Document
to OpenGL, Version 1.1

Industry Standard 3D Technology

OpenGL[®] Programming Guide

Third Edition
The Official Guide to Learning
OpenGL, Version 1.2



NVIDIA

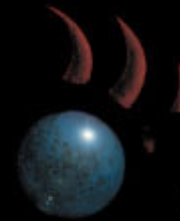




Demo



Not Just for Games



kaydara



Alias | **wavefront**





Demo

Mac OS X OpenGL

- State-of-the-art architecture
 - Resource virtualization
 - Data flow optimizations
 - Overlays through semi-transparent windows
- Apple co-develops drivers
 - Single stop shopping for developer issues
 - Can respond quickly to developer requests
 - Maintain consistence across product line



OS X OpenGL

- Apple has good visibility into 3D hardware
 - Apple can help push your requests into hardware
 - Apple can find unique solutions to your needs
- Apple taking a leadership role in OpenGL ARB



What's New in OpenGL

- Programmable Shaders
- System Integration
- Texture upload performance
- Lots of extensions
- Amazing new Tools



Programmable Shaders

- Vertex Shading based on:

GL_ARB_vertex_program

- Pixel and Texture shading based on vendor-specific extensions:

GL_NV_texture_shader[123]

GL_NV_register_combiners[12]

GL_ATIX_fragment_program



OpenGL Shader Builder

The screenshot displays the OpenGL Shader Builder interface. The main window is titled "OpenGL Shader Builder" and contains a code editor for a vertex shader named "Lighting.shdr". The code is written in ARB Vertex Program language and includes attributes for position, normal, color, and texture coordinates, as well as matrices for model-view and projection. It defines a light direction vector and performs calculations for output position, color, and texture coordinates.

```
Language ARB Vertex Program | Instruction Reference

ATTRIB vPosition = VERTEX POSITION;
ATTRIB vNormal   = VERTEX NORMAL;
ATTRIB vColor    = VERTEX COLOR;
ATTRIB vTexCoord = VERTEX_TEX_COORD, 0;

PARAM m MVP [4] = { MATRIX_MODELVIEW_PROJECTION, IDENTITY };
PARAM mt MVP [4] = { MATRIX_MODELVIEW_PROJECTION, INVERSE_TRANSPOSE };

PARAM lightDirection = { -0.2, -0.2, -0.2, 1.0 };

RESULT outPosition = OUTPUT POSITION;
RESULT outColor    = OUTPUT COLOR, FRONT;
RESULT outTexCoord = OUTPUT_TEX_COORD, 0;

TEMP temp0, nPrime;

MMULT outPosition, m MVP, vPosition; # Output Position

MMULT nPrime, mt MVP, vNormal;

DP3 temp0, lightDirection, nPrime;
MUL temp0, temp0, vColor;
SWZ outColor, temp0, x.y.z.1;

Code OK
```

The 3D preview window shows a rendered scene of a ship, with a coordinate system (r.x, r.y, r.z) and a "Ship" label. Below the preview is a "GL Parameters" table:

Identifier	Value
vPosition	[0.00, 0.00, 0.00, 0.00]
vNormal	[0.00, 0.00, 0.00, 0.00]
vColor	[0.00, 0.00, 0.00, 0.00]
vTexCoord	[0.00, 0.00, 0.00, 0.00]
m MVP [0]	[0.00, 0.00, 0.00, 0.00]
m MVP [1]	[0.00, 0.00, 0.00, 0.00]
m MVP [2]	[0.00, 0.00, 0.00, 0.00]
m MVP [3]	[0.00, 0.00, 0.00, 0.00]
mt MVP [0]	[0.00, 0.00, 0.00, 0.00]
mt MVP [1]	[0.00, 0.00, 0.00, 0.00]
mt MVP [2]	[0.00, 0.00, 0.00, 0.00]
mt MVP [3]	[0.00, 0.00, 0.00, 0.00]
LightDirec	[-0.20, -0.20, -0.20, 1.00]
outPosition	[0.00, 0.00, 0.00, 0.00]
outColor	[0.00, 0.00, 0.00, 0.00]
outTexCoord	[0.00, 0.00, 0.00, 0.00]
temp0	[0.00, 0.00, 0.00, 0.00]
nPrime	[0.00, 0.00, 0.00, 0.00]

The right sidebar contains a "Double click to insert instruction into code" list of operations:

- Op Description
- ARL Address Register Load
- MV Move
- LIT Compute lighting values
- RCF Reciprocal
- RSQ Reciprocal Square Root
- EXP Base 2 Exponential
- LOG Base 2 Logarithm
- FRC Fractional Component
- MUL Multiply
- ADD Add
- DP3 Three component dot product
- DP4 Four component dot product
- DST Compute distance vector
- MIN Minimum value
- MAX Maximum value
- SLT Set less than
- SE Set greater than or equal to
- XPD Cross product
- MAD Multiply Add
- CLP Clamp
- SWZ Extended Swizzle

The "SLT - Set less than" section is expanded, showing the following description:

The set less than operation performs a component wise compare of its sources and writes a 1 into each component of the destination where the first source was less than the second source for that component. For all other cases, the destination component is set to 0. This operation is a binary operation, so it takes two source variables. The operation executes as follows:

```
if SrcA.x < SrcB.x then
  Dst.x = 1.0
else
  Dst.x = 0.0

if SrcA.y < SrcB.y then
  Dst.y = 1.0
else
  Dst.y = 0.0

if SrcA.z < SrcB.z then
  Dst.z = 1.0
else
  Dst.z = 0.0

if SrcA.w < SrcB.w then
  Dst.w = 1.0
else
  Dst.w = 0.0
```

The bottom of the window features a "Debugger" section with "Run", "Step", and "Reset" buttons.



System Integration

- Quartz on a texture
 - High quality 2D anti-aliased content on ARGB textures
- Video on a texture
 - Non power of 2 textures
 - YUV texture formats
- Window Compositor integration
 - Overlays on steroids
 - “Heads up” UI for 3d apps



Texture Upload Performance

- Direct DMA from native texture formats
- No per pixel CPU involvement
- Flexible synchronization model
 - **GL_APPLE_fence**



New Extensions in Jaguar

GL_APPLE_ycbcr_422
GL_APPLE_texture_range
GL_APPLE_fence
GL_APPLE_vertex_array_range
GL_APPLE_vertex_array_object
GL_ARB_imaging
GL_ARB_texture_env_crossbar
GL_ARB_multisample
GL_ARB_point_parameters
GL_ARB_pipeline_program
GL_ARB_vertex_program
GL_ARB_texture_mirrored_repeat
GL_EXT_secondary_color
GL_EXT_fog_coord
GL_EXT_draw_range_elements
GL_EXT_stencil_wrap
GL_EXT_blend_func_separate

GL_SGIX_depth_texture
GL_SGIX_shadow
GL_ATI_texture_mirror_once
GL_ATIX_fragment_program
GL_ATI_blend_equation_separate
GL_ATI_blend_weighted_minmax
GL_NV_blend_square
GL_NV_fog_distance
GL_NV_multisample_filter_hint
GL_NV_point_sprite
GL_NV_texture_shader
GL_NV_texture_shader2
GL_NV_texture_shader3
GL_NV_depth_clamp



OpenGL Profiler

The screenshot displays the OpenGL Profiler application interface, which is used for analyzing the performance of OpenGL applications. The main window is titled "OpenGL Profiler" and contains several panels:

- Chess Game:** A 3D chess game is running in the background, showing a chessboard with pieces.
- Breakpoints:** A list of GL functions with checkboxes for "Break Before", "Break After", and "Profile".
- GL State:** A table showing the current state of various GL parameters, such as window bounds, texture coordinates, and active textures.
- OpenGL Application:** A panel for managing the application being profiled, including the application name ("chess"), launch arguments, and the full path.
- OpenGL Profile:** A panel for configuring the profiler's behavior, including "Debug Mode", "Function Stats", "Function Trace", "Force Flush", "Frame Rate", and "No Graphics".
- OpenGL Drivers:** A panel for selecting the driver to track, currently set to "Track Application".
- OpenGL Function Statistics:** A table showing the total time, number of calls, average time, and percentage of total time for various GL functions.
- OpenGL Function Trace:** A list of GL functions with their arguments, used for tracing the execution of the application.
- OpenGL Driver Monitor:** A graph showing the performance of various driver parameters over time, with a horizontal scale of 5.09.

The bottom status bar shows the total elapsed time: 69678757.72 usec. The context ID is "All contexts".

GL Function	Total Time (usec)	Number of Calls	Average Time (usec)	% Total Time
glCallList	6708558.99	154,494	434.23	96.27835109
glVertex3f	989085.54	721,028	1.37	1.41949365
glBegin	303527.92	14,046	21.61	0.434561041
glTranslatef	302514.18	208,555	1.45	0.43415552
glScalef	261868.95	156,835	1.67	0.37582322
glRotatef	220373.87	70,663	3.12	0.31627124
glPopMatrix	78924.41	163,858	0.48	0.11326897
glClear	61645.14	2,341	26.33	0.08847049
glTexCoord2f	52619.17	711,664	0.07	0.07551680
glPushMatrix	51764.21	163,858	0.32	0.07431850
glEnd	41890.80	14,046	2.98	0.06011989
glMaterialfv	38802.48	35,115	1.11	0.05568768
glDisable	30047.53	21,069	1.43	0.04312295
glLightfv	29134.43	7,023	4.15	0.04181250
glNormal3f	28613.60	25,751	1.11	0.04106503
glMaterialf	24003.58	11,705	2.05	0.03444892
glBindTexture	20653.51	7,023	2.94	0.02964105
glEnable	16469.88	21,069	0.78	0.02363687
glLoadIdentity	9221.89	2,341	3.94	0.01323486
glMultMatrixd	8543.29	2,341	3.65	0.01226097
glClipPlane	7758.03	4,682	1.66	0.01113400
glStencilOp	4355.32	4,682	0.93	0.00625057
glColor4f	3463.28	2,341	1.48	0.00497036
glStencilFunc	2725.47	4,682	0.58	0.00391148
glBlendFunc	2501.92	2,341	1.07	0.00359065
glColorMask	1675.57	2,341	0.72	0.00240471
glTranslated	994.75	2,341	0.42	0.00142762
CGLFlushDrawable	0.00	2,341	0.00	0.00000000
CGLUpdateContext	0.00	2,341	0.00	0.00000000
CGLGetSurface	0.00	2,341	0.00	0.00000000



Open GL Roadmap

**504 OpenGL: Graphics
Programmability**

Room A2
Tue., 5:00pm

505 OpenGL: Integrated Graphics 1
Putting OpenGL to work for you

Room J
Wed., 9:00am

506 OpenGL: Integrated Graphics 2
Putting OpenGL to work for you

Room J
Wed., 10:30am

513 OpenGL: Advanced 3D
Latest extensions and techniques

Room J
Thurs., 3:30pm

**514 OpenGL: Performance
Optimization**

Room J
Thurs., 5:00pm



Aqua

Frameworks

Quartz

OpenGL

QuickTime

Darwin



QuickTime Roadmap

600 The State of QuickTime in 2002

Room A2
Wed., 9:00am

601 Building QuickTime Savvy Apps

Room A2
Wed., 10:30am

602 QuickTime for Video-Intensive Applications

Room A2
Wed., 2:00pm

603 Media Integration with QuickTime

Room A2
Wed., 3:30pm

604 Delivering Content via Interactive QuickTime

Room A2
Wed., 5:00pm



QuickTime Roadmap (Cont.)

FF010 Feedback Forum: QuickTime

Room J1
Fri., 10:30am

606 QuickTime for the Web

Room A2
Fri., 2:00pm

**607 QuickTime and MPEG4:
A Technical Overview**

Room A2
Fri., 3:30pm



Aqua

Frameworks

Quartz

OpenGL

QuickTime

Darwin



Quartz

OpenGL

QuickTime

Quartz Compositor



Window System as a Digital Image Compositor

COMPUTER GRAPHICS

Volume 18 • Number 3 • July 1984
A quarterly report of ACM SIGGRAPH

SIGGRAPH '84 Conference Proceedings
July 23-27, 1984, Minneapolis, Minnesota
Edited by Hank Christiansen

Compositing Digital Images

Thomas Porter
Tom Duff†

Computer Graphics Project
Lucasfilm Ltd.

ABSTRACT

Most computer graphics pictures have been computed all at once, so that the rendering program takes care of all computations relating to the overlap of objects. There are several applications, however, where elements must be rendered separately, relying on compositing techniques for the anti-aliased accumulation of the full image. This paper presents the case for four-channel pictures, demonstrating that a matte component can be computed similarly to the color channels. The paper discusses guidelines for the generation of elements and the arithmetic for their arbitrary compositing.

CR Categories and Subject Descriptors: 1.3.3 [Computer Graphics]: Pictures/Image Generations — Display algorithms; 1.3.4 [Computer Graphics]: Graphics Utilities — Software support; 1.4.1 [Image Processing]: Digitization — Sampling.

General Terms: Algorithms

Additional Key Words and Phrases: compositing, matte channel, matte algebra, visible surface algorithms, graphics systems

1. Introduction

Increasingly, we find that a complex three dimensional scene cannot be fully rendered by a single program. A wealth of literature on rendering polygons and curved surfaces, handling the special cases of fractals and splines and quadrics and triangles, implementing refinement, texture mapping and bump mapping, noting speed-ups on the basis of coherence or depth complexity in the scene suggests that multiple programs are necessary.

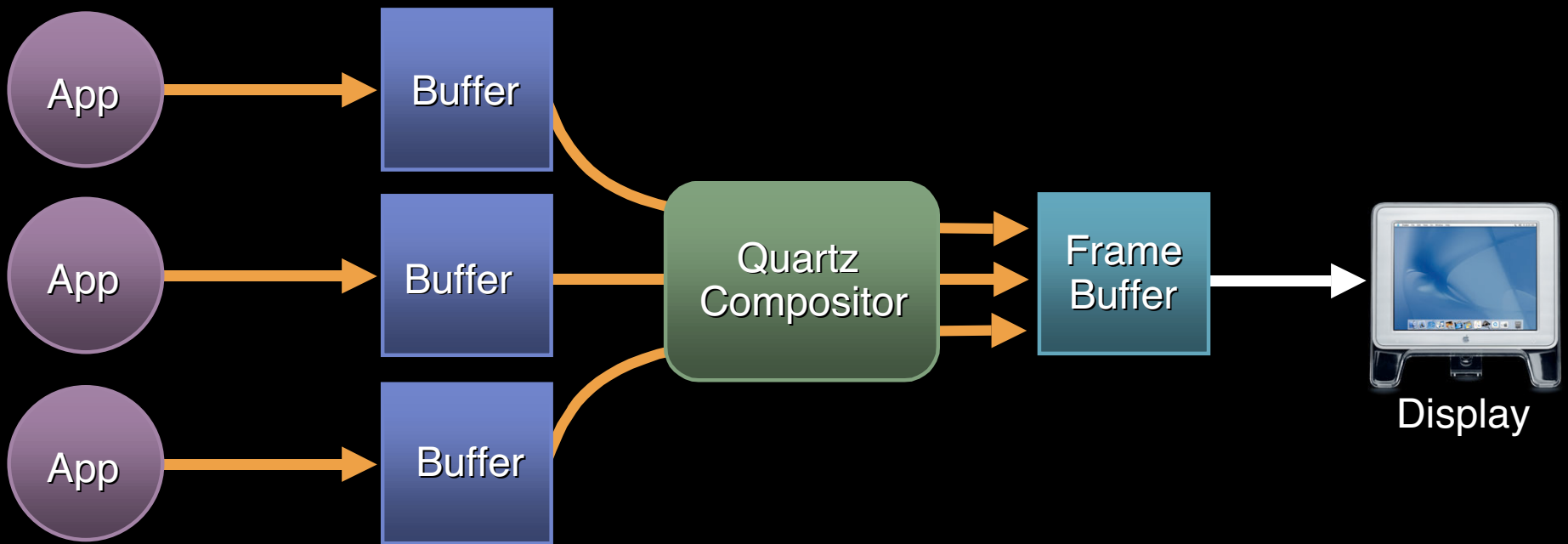
In fact, reliance on a single program for rendering an entire scene is a poor strategy for minimizing the cost of small modeling errors. Experience has taught us to break down large bodies of source code into separate modules in order to save compilation time. An error in one routine forces only the recompilation of its module and the relatively quick reloading of the entire program. Similarly, small errors in coloration or design in one object should not force the "recompilation" of an entire image.

Separating the image into elements which can be independently rendered saves enormous time. Each element has an associated matte, coverage information, which designates the...



Demo

Quartz Compositor



But . . .

- Model can be computationally expensive
- “You took the framebuffer away . . .”
- We’ve built the foundation for the future
- Now we can complete the story . . .





Quartz Extreme



- Implementation of Quartz Compositor on OpenGL
- Removes “transparency tax” for video and 3D
- Frees up CPU
- Showcases GPU in user interface
- Allows us to deliver even more dramatic UI advances





Demo

Programmed I/O vs. DMA

- Programmed I/O Model
 - CPU pushes data and commands to device
 - Inefficient use of CPU
- DMA I/O Model
 - Device pulls data and commands from memory
 - CPU and I/O occur in parallel
- CPU drawing in the framebuffer is really just programmed I/O!



More Moore's Law

- CPU

- Performance doubles every 18 months
- G4: 10 million transistors

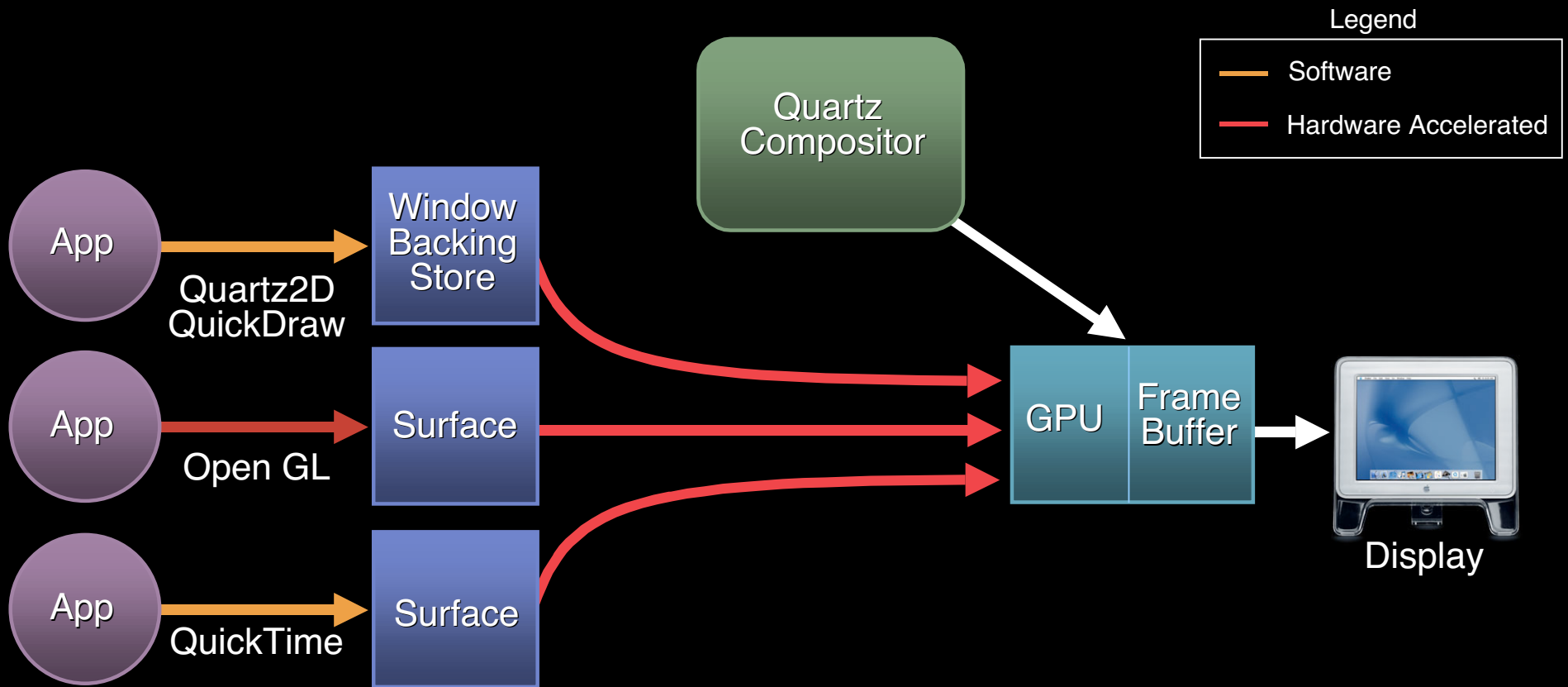


- GPU

- Performance doubles every 6 months
- “Moore’s law cubed”
- GeForce4 Ti: 63 million transistors



Quartz Extreme



Apple Leading the Industry

- Natural evolution of windowing systems
- Lots of hacks out there, but we've done it right
- Inflection point in platform graphics architecture
- Our advances are directly useable by you via Jaguar OpenGL



Quartz Compositor Roadmap

503 Exploring the Quartz Compositor

Hall 2
Tue., 3:30pm

506 OpenGL: Integrated Graphics 2

Build Your Own Compositor!

Room J
Wed., 10:30am



Aqua

Frameworks

Quartz

OpenGL

QuickTime

Quartz Compositor

Darwin



Quartz

OpenGL

QuickTime

Quartz Compositor



Quartz

OpenGL

QuickTime

Quartz Compositor

ColorSync

**Image
Capture**

Printing



Quartz

OpenGL

QuickTime

Quartz Compositor

ColorSync

**Image
Capture**

Printing



ColorSync

- ICC standard color matching
 - www.color.org
- Framework for color calculations
- Device transforms expressed as profiles
- Built into Quartz 2D
- Successfully used in print publishing
- Emerging for film and video



What's New in ColorSync

- Tuned for Velocity Engine
- ICC4
- Convenience Colorspaces for Quartz 2D
- Realtime RGBtoRGBA possible on GeForce4 Ti





Demo

ColorSync Roadmap

509 ColorSync and Digital Media:

Room C
Wed., 5:00pm



Quartz

OpenGL

QuickTime

Quartz Compositor

ColorSync

**Image
Capture**

Printing



Image Capture Workflow

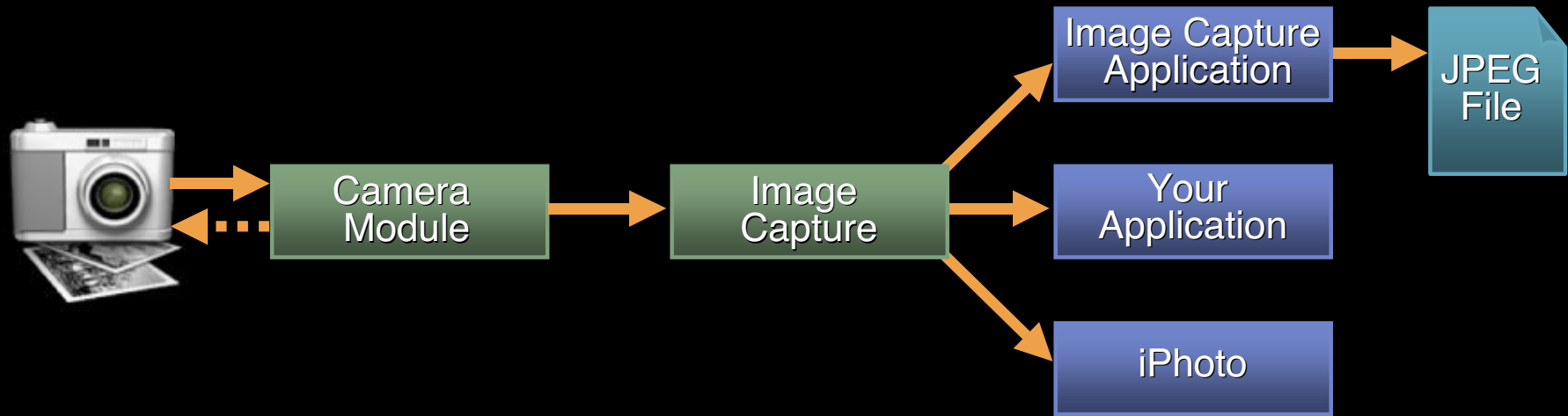
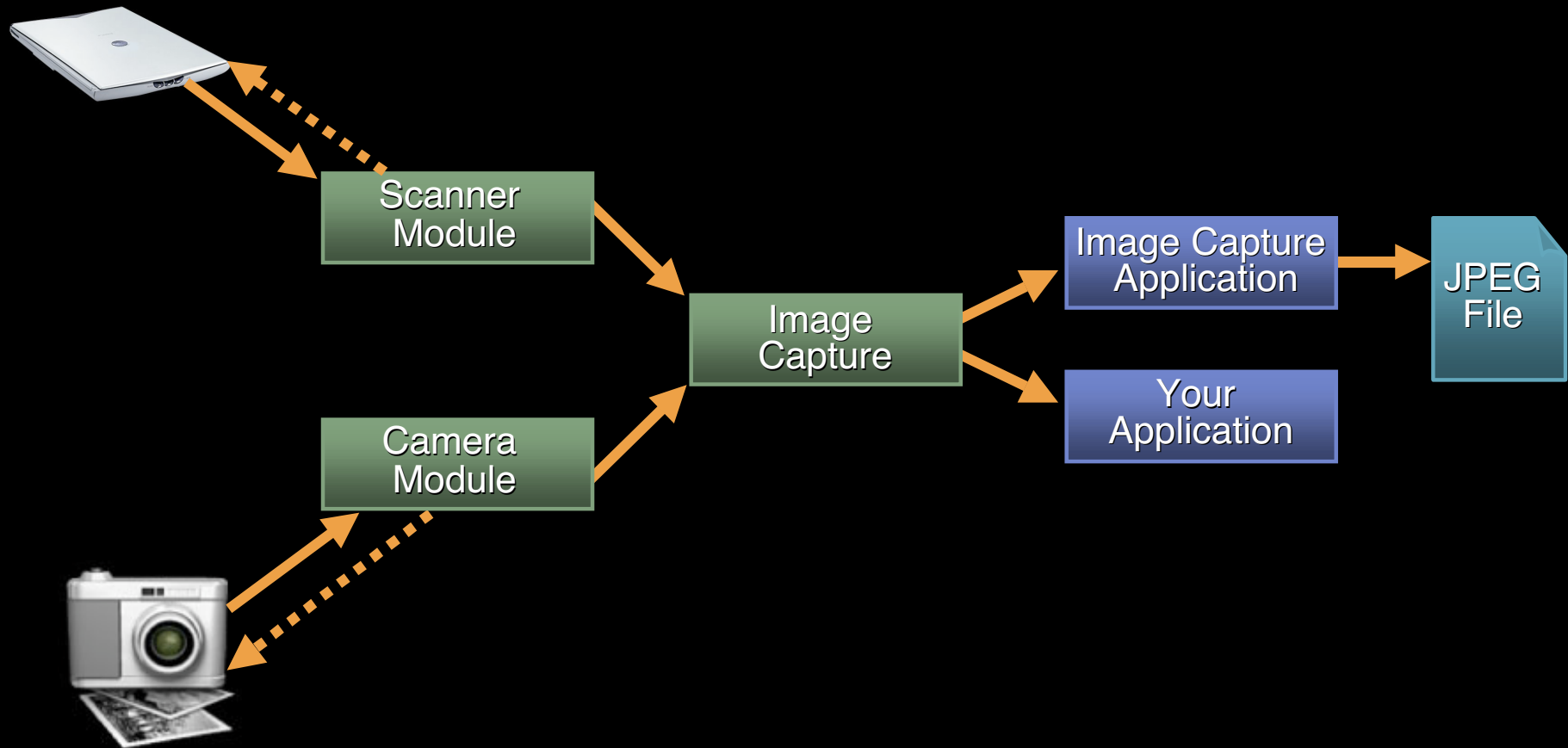


Image Capture Workflow



What's New in Image Capture

- Scanner support
- TWAIN
- Basic scanning UI
- FireWire camera support



Built-in Scanner UI



Image Capture Roadmap

515 Image Capture Framework

Room C
Fri., 2:00pm



Quartz

OpenGL

QuickTime

Quartz Compositor

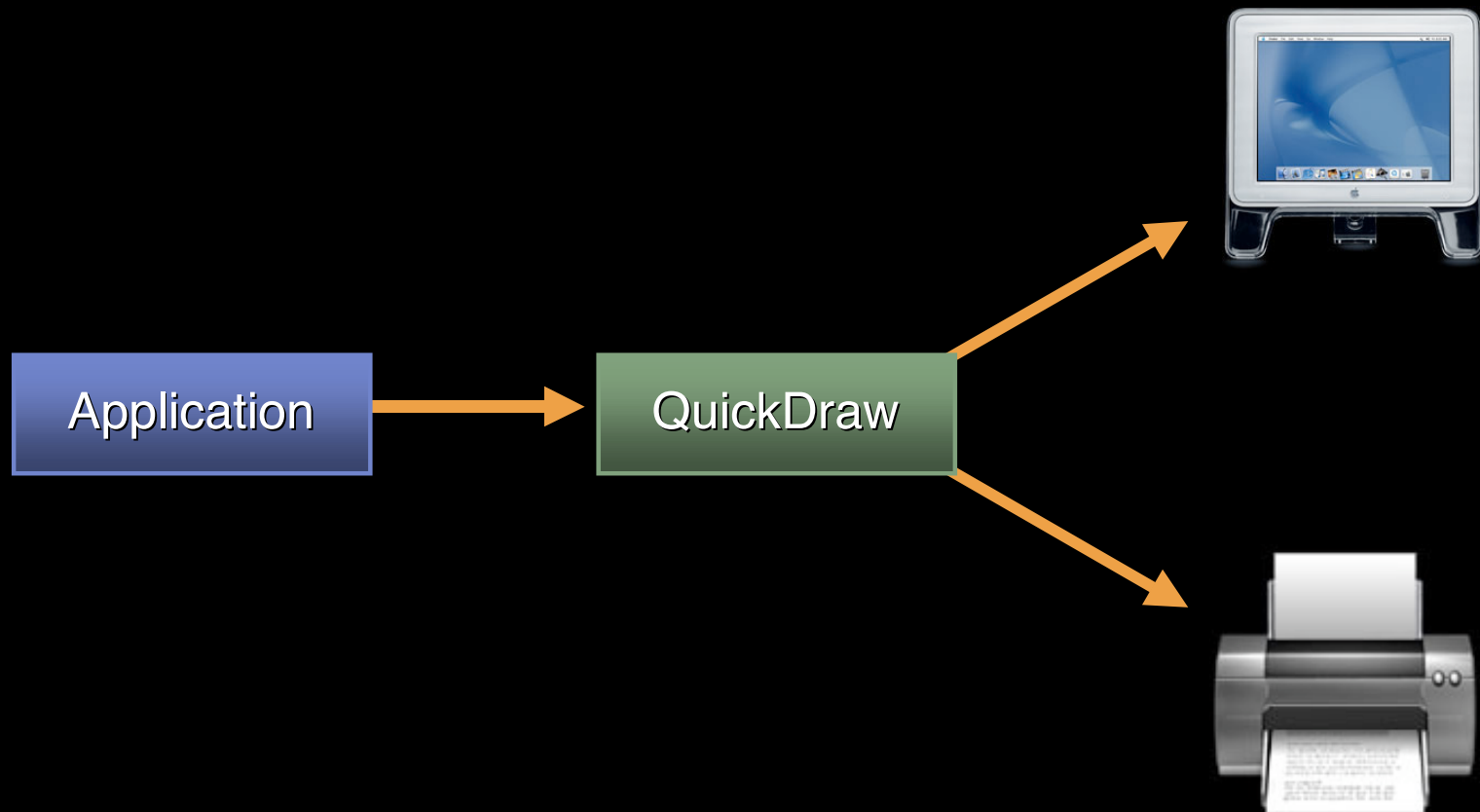
ColorSync

**Image
Capture**

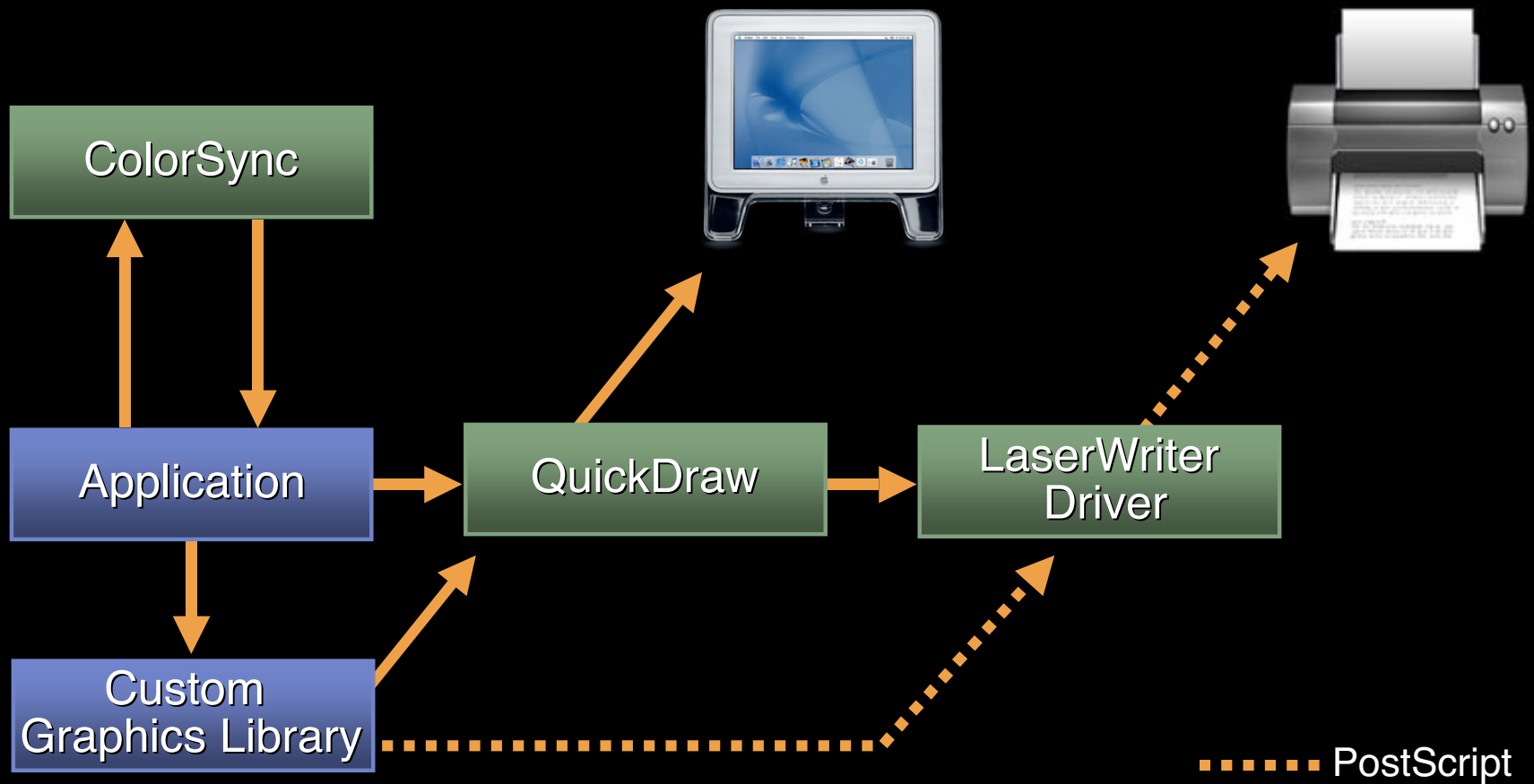
Printing



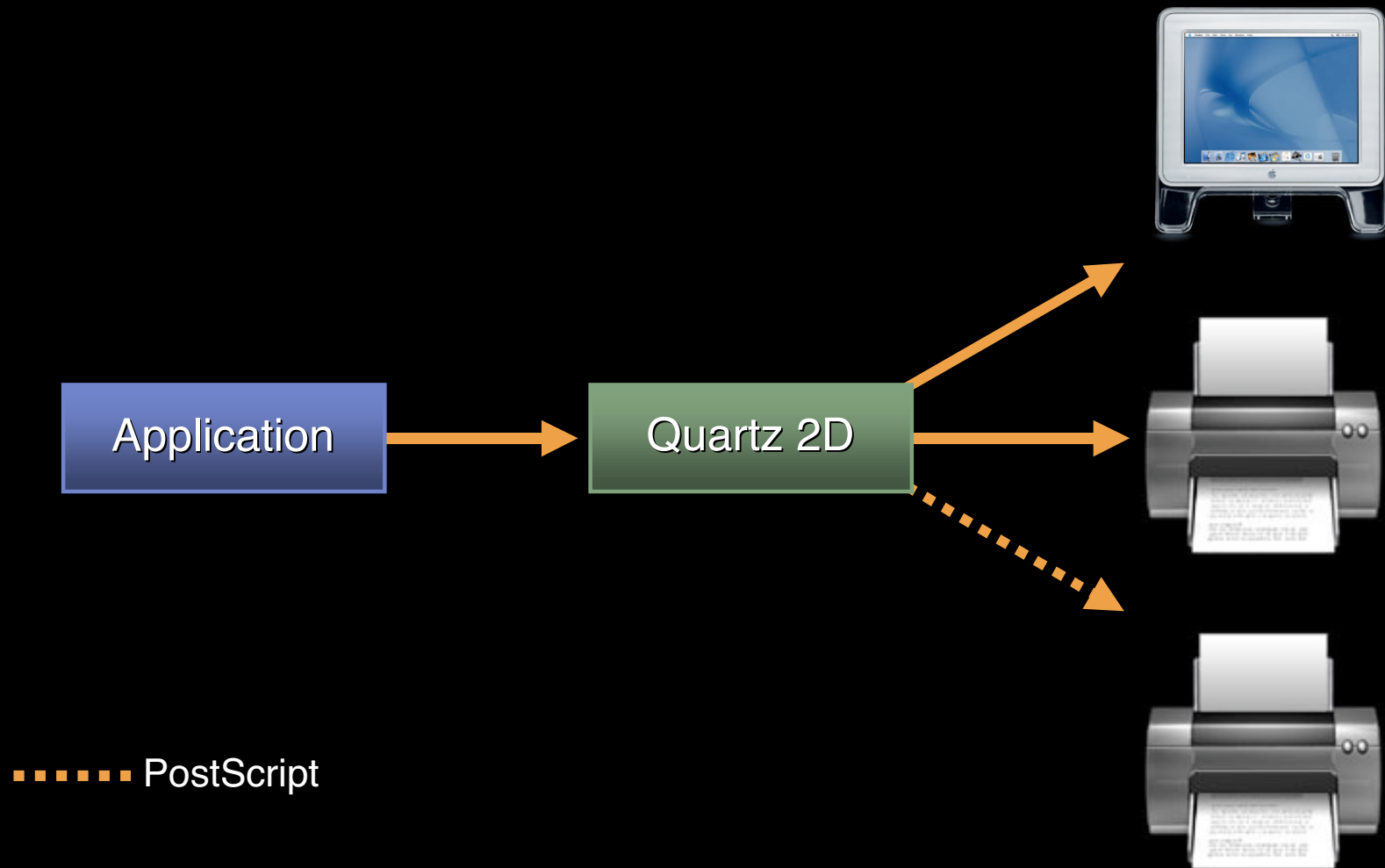
The Beginning: QuickDraw Does It All



The Reality: Postscript Was Required



Today: Printing With Quartz 2D





Quartz 2D



Quartz 2D



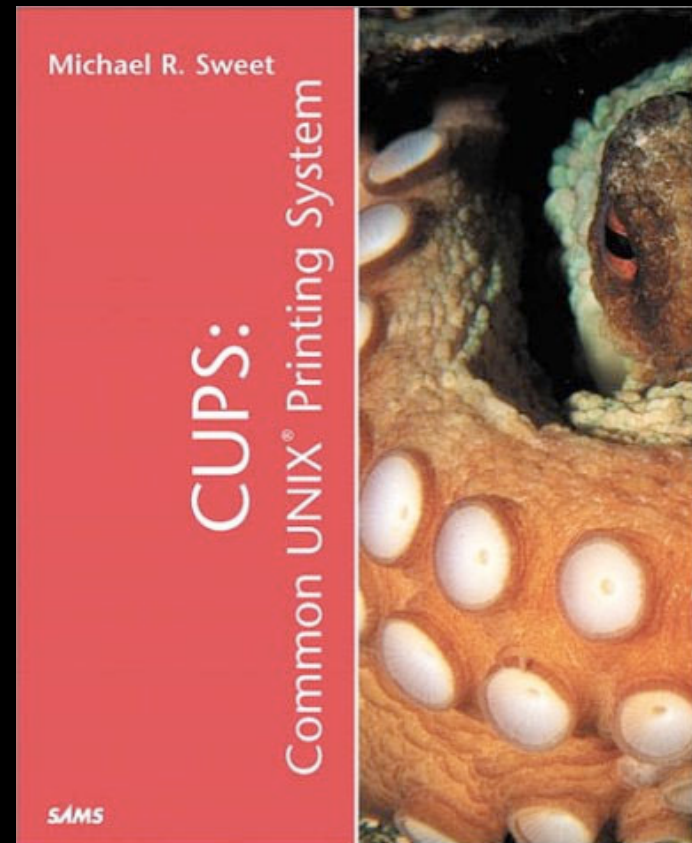
What's New in Printing

- Completely new OpenSource spooling system
- Binary compatible with 10.1 for Apps
- Binary compatible with 10.1 for drivers too!
- New features
 - Printer sharing via IPP
 - Cocoa Print Center



Common UNIX Printing System

- Printer Sharing via IPP
- UNIX lpr/lp suite
- Flexible architecture
- In several Linux distros
- www.cups.org





Demo

Printing Roadmap

510 Printing and Mac OS X:
APIs and Architecture Update

Hall 2
Thurs., 10:30am

109 Darwin Printing:
All About CUPS From Michael Sweet

Room J
Wed., 2:00pm



Doc in the House

New documentation since WWDC '01

- 1000's of pages of new or revised documentation
 - 8+ revised books
 - 38 new Q&As
 - 4 new Technotes
 - 78 new code samples
 - Full set of OpenGL man pages

Feedback

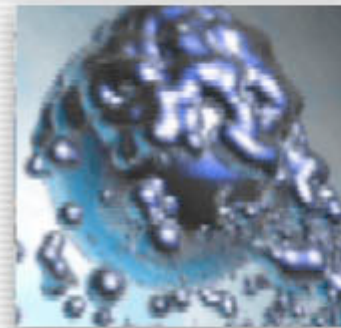
FF018 Graphics and Imaging

Room J1
Fri., 5:00pm





Q&A



Travis Brown
Graphics and Imaging Evangelist
Worldwide Developer Relations

<http://developer.apple.com/wwdc2002/urls.html>

 **WWDC2002**

 **WWDC2002**

 **WWDC2002**